Integrating BOINC with Microsoft Excel: A Case Study

Jingri Zhang, Navonil Mustafee, Jon Saville and Simon J E Taylor
Centre for Applied Simulation Modelling
Brunel University, Uxbridge, UB8 3PH, UK
{ci05jjz,navonil.mustafee,jon.saville,simon.taylor}@brunel.ac.uk

Abstract. *The convergence of conventional Grid computing with public resource computing (PRC) offers potential benefits in the enterprise setting. For this work we took the popular PRC toolkit BOINC and used it to execute a previously monolithic Microsoft Excel financial model across several commodity computers. Our experience indicates that speedup approaching linear may be realised for certain scenarios, and that this approach offers a viable route to leveraging idle desktop PCs in the enterprise.*

Keywords. BOINC, Excel, Public Resource Computing, Desktop Grids.

1. Introduction

Various new approaches to distributed computing have arisen in recent years. Although originally targeted at different application domains, there are inevitably areas of overlap in function and terminology [10].

We outline a hybrid approach known as 'enterprise desktop grids' (EDGs) [7] [12] [13], which potentially offers the benefits of both standard Grid computing [9] and public resource computing (PRC) [2] when employed within an enterprise to support and accelerate their applications.

This paper sets out to investigate the potential benefits of this approach, and is structured as follows: in Section 2 we discuss the background to EDGs, and the PRC toolkit we have chosen to explore this work, BOINC. Section 3 considers BOINC and enterprise computing. Section 4 outlines a case study that illustrates how BOINC can be deployed in an enterprise setting. Section 5 discusses the results of experimentation with the case study. Section 6 concludes the paper.

2. Grid and Public Resource Computing

Foster and Kesselman define Grid computing [9] as an approach to providing access to significant computing resources on demand. Users expect professionally managed, high availability, typically cluster-based resources, and a rich set of standardised services from which to construct distributed applications.

In contrast, PRC, also known as volunteer computing [5], harnesses idle CPU cycles on large numbers of relatively low-powered desktop computers to provide considerable aggregate computing resources. PRC projects typically use bespoke software toolkits optimised to implement the minimum number of distributed services required to achieve the project at hand. The type of project which most benefits from this approach is one contributing to the public good (thus attracting volunteers), and having a problem domain which may be partitioned with fine granularity (in order to execute on the volunteers' computers in a timely fashion). Since the providers of computing resource are numerous, not secure, and only minimally authenticated, PRC is not appropriate for processing proprietary data. Additional measures are necessary to ensure that work units are not lost, and that results are not fabricated. The Berkeley Open Infrastructure for Network Computing (BOINC) [1] is perhaps the most well known toolkit for PRC.

A hybrid approach known as desktop Grid computing potentially offers the advantages of both approaches [7]. By confining access to desktop computers and users within the same organisation, much of the complexity and security overhead of conventional Grid computing is avoided. Although less aggregate computing power is available than with either of the other two approaches, it is sufficient to benefit many applications that would otherwise be intractable on a single machine. Similarly, the restricted extent of the system (within the firewall of an organisation or department) allows proprietary data to be processed, and ownership of the computing resources ensures that user cooperation and execution times are more predictable than with PRC.

2.1. BOINC

Although BOINC was originally designed only to support the PRC form of distributed computing, of late there has been a realisation that the same software can be reconfigured to support desktop Grid computing.

BOINC was developed by those responsible for the PRC project SETI@home [4], which originally used bespoke software to search for evidence of extraterrestrial intelligence in radio signals. BOINC now provides a generic set of tools and patterns which are used to support a wide range of PRC projects. Presently, BOINC is used by around 20 such projects, which together consume an estimated 350 teraflops of processing power, generated by approximately 1 million computers contributed by some 600,000 volunteers [3]. Areas of study include mathematics, medicine and the physical sciences (for example [11] [15] [16]).

The BOINC system contains several server-side components, which may execute on separate machines if required. A database holds all the metadata associated with the project, and lifecycle information for each work unit. A client's command channel operates via a scheduling server, using an XML-based protocol. Results are transferred using HTTP via data servers. In addition to work units and results, other files may be transferred between server and client, including application executables and any other interim data the application may require during operation. On the client side, the BOINC client engine manages interaction with the server, while optional components provide graphical control and display elements for the benefit of the user. A client API provides the interface between the application and BOINC, and is linked with the application executable.

BOINC includes many other features which are not relevant in the context of EDGs, including community and reward features, anti-cheating measures, and mechanisms to deal with client or network failure.

3. BOINC in an enterprise setting

Ethernet networking technologies are pervasive in most enterprise settings. However, individual departments may retain separate policies regarding usage of their computing resources, and will typically install different sets of applications on to their workplace computers. Similarly, although a single BOINC server can support multiple projects, individual departments may wish to manage their own BOINC installation. The choice of which computers across the enterprise will participate will depend on departmental goals and responsibilities, in addition to technological factors.

3.1. Types of BOINC application

In the PRC setting, applications are downloaded from the server by the BOINC client as required. Only BOINC itself need be pre-installed on each client computer. We refer to this type of BOINC application as 'runtime', because there are no client-side dependencies; this may in turn encourage participation outside of the project sponsor's department. A disadvantage is the need to package applications in the downloadable form that BOINC requires, which may require development work.

Within the enterprise, the applications we wish to utilise are often installed on the client computers already, particularly if they form part of an office productivity suite. We call this type of BOINC application 'pre-installed'. The advantage here is that only a small application proxy is downloaded to the client, which is used by BOINC to call the pre-installed application. There may be additional administration overheads, however, such as ensuring that security permissions and application versions are correct on every participating client machine.

Applications vary widely in their installed footprint, size of work unit, and disk and memory space needed during execution [8]. The choice of runtime versus pre-installed approach will depend on these practical factors as well as the administrative policies in place.

We have previously investigated the pre-installed approach using a non-BOINC desktop grid [13] [14], but the focus in this paper is on pre-installed BOINC applications.

3.2. Departmental participation

In a BOINC-based desktop grid environment the inter-departmental participation in a project may vary depending on whether it is a runtime or pre-installed BOINC application. It is relatively easy for different departments to participate in projects that do not impose any client side dependencies. However this camaraderie may not always be possible if BOINC projects require invocation of third-party applications which first have to be installed to client PCs.

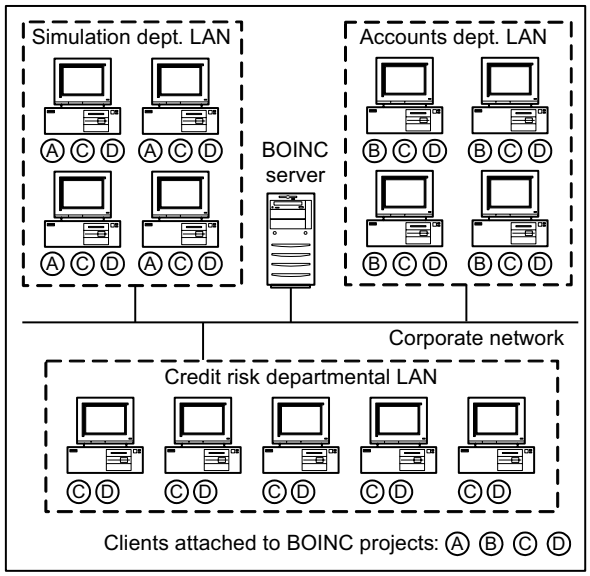

Figure 1: Multiple BOINC projects

For example, the simulation department may create BOINC project 'A' using a specialist software package. But the accounts department in the same organisation may not be able to participate in such simulation projects because their departmental PCs are only installed with specialist financial software. They can, however, create a BOINC project 'B' to handle their own processing requirements. The credit risk department may create BOINC project 'C' that requires Microsoft Excel. Since Excel is a widely used application we can expect it to be installed on most PCs in the organisation. Thus the simulation and the accounts departments can join in with the finance department to execute the Excel-dependent BOINC application on their respective departmental resources. Similarly, a runtime BOINC application (project 'D') created by the accounts department can be easily executed by all three departments due to the lack of client-side dependencies. Figure 1 shows these four different BOINC execution scenarios.

4. Case study

To investigate how BOINC can be used in a desktop grid environment, where the execution of the BOINC application is reliant on pre-installed software on the client computers, we experimented with a Microsoft Excel based Monte Carlo simulation application used for financial modelling by a leading European financial institution. This financial model calculates the risk of a 'Range Accrual Swap' at various points in time until the maturity of the transactions, using a Monte Carlo process.

Successful and accurate calculation of risk on the Range Accrual Swap requires a large number of Monte Carlo simulations and takes a significant amount of time. Each simulation run (iteration) is independent of previous runs and is characterised by the generation of random values for various defined variables, and solving equations containing those variables. The current approach of using only one instance of Microsoft Excel is not feasible in situations where the business desires a quick turnaround. One solution to this is to distribute the processing of the model among a network of computers by utilising their spare processing power and the Excel software installed on them. Thus, if the Range Accrual Swap model requires 100,000 iterations and we have 10 computers at our disposal, we could assign each computer 10,000 iterations. In order to arrive at the final values we have only to take the arithmetic mean of the result sets (10 in this case). This distributed approach has the potential of speeding up the execution of the financial model many fold.

4.1. The BOINC application

The BOINC application for this experiment consists of server-side administrative components, and client-side components designed to interface between BOINC and Excel.

On the client side (figure 2), our solution comprises a BOINC client application proxy written in Visual C++, a Dynamic Link Library (DLL) written in Visual Basic, and the Excel spreadsheet itself. The client application proxy is the item executed by BOINC to perform work, and it interfaces directly with the BOINC client API. It manages the lifecycle of the financial model, and the various files consumed or produced during the simulation. The DLL abstracts the control interface to Excel, allowing an application to perform basic operations such as opening and closing spreadsheet files, starting the simulation iterations and so on. The DLL achieves this by using the COM interface to Excel, and as a result the client application proxy can fully automate Excel's operation.

The number of iterations to be performed for each execution of the simulation is read at runtime from a parameter file (parameter.txt). Although this file allows considerable control over the sequence of model executions, for consistency in our experiments we specified 300 iterations for each run.

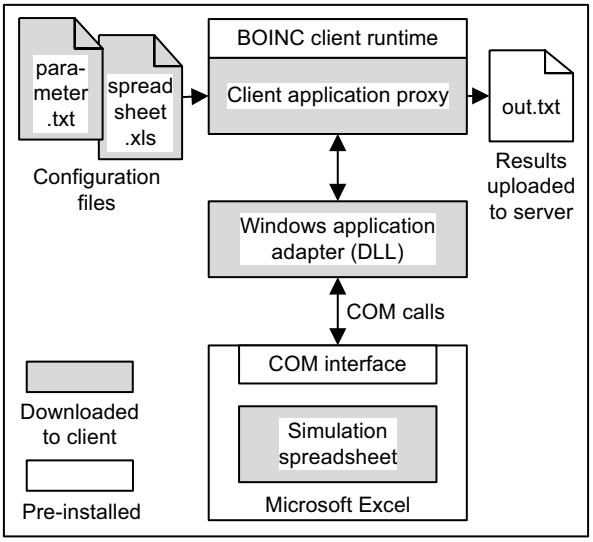

Figure 2: Excel client architecture

On the server side, work units must be created for BOINC to distribute to waiting clients. In our case, work units contain only metadata, and were automatically generated by calling the BOINC `create_work` command from a custom Java application. `create_work` requires template files which specify the format of work units and results. For this application, the template also includes details of the three components to be downloaded to each client.

Since we are using BOINC in an enterprise environment, we assume that client computers can be trusted (i.e., the operation of the client will not be interfered with to produce incorrect results). As such, we do away with redundant processing of work units, and create only one instance for every work unit. Thus, the minimum size of quorum [6] required to validate the result, and the maximum number of results allowed for a work unit, are both set to 1 in the template file.

When the BOINC client first attaches itself to our project, it downloads from the BOINC server the client components, configuration files, and a number of project work units. Results are accrued into a plain text file (out.txt) during execution, which is uploaded to the server on completion of the simulation.

This process is repeated on each of the BOINC clients, which connect independently to the server to acquire further work units.

4.2. The BOINC test network

Our BOINC project runs within the confines of the Brunel University firewall and comprises:

(1) One PC running the CentOS 4.3 operating system with a 864MHz Pentium III processor and 256MB RAM. All BOINC server-side components and the MySQL database are executed here.

(2) Four laptop computers running the Microsoft Windows XP Professional operating system, each with a 1.73GHz Intel Celeron processors and 1GB RAM. Each laptop is pre-installed with the BOINC client and Microsoft Excel. One of these laptops has two network cards, and acts as the router between the test LAN and the BOINC server.

(3) Four low-end desktop PCs running either Microsoft Windows XP Professional or Microsoft Windows 2000, with either Pentium I or Pentium II processors and 128MB or 256MB RAM. Like the laptops, they are pre-installed with Excel and BOINC clients.

(4) The laptops and desktop PCs are connected through a 100Mbps switch to form a private test LAN.

5. Results and discussion

Our experiment consisted of timing the execution of 50, 100, 150 and 200 work units of our Excel-based Monte Carlo financial simulations. We compared the execution time obtained from our distributed BOINC implementation with that of the simulation running in a standalone fashion on one of the laptops equipped with a 1.73GHz Intel Celeron processor and 1GB RAM.

The results are summarized in figure 3, which shows execution time per work unit, averaged over five separate runs of the experiment. However, this graph only includes experiments using the four laptop BOINC clients, for reasons outlined below.

The graph shows that the speedup is approximately linear compared to standalone execution for the range of workloads we tested. This was expected for several reasons: client computers were entirely dedicated to running the simulation; work units carried little data due to the nature of the simulation; the BOINC client pre-fetches new work units from the server so that it may continue uninterrupted. Under these circumstances, BOINC imposes very little overhead.

Our use of client computers that were entirely dedicated to running the simulation may not reflect the reality of daytime use of computers in the workplace, where BOINC typically executes in the background. Many organisations, however, need to execute this type of application

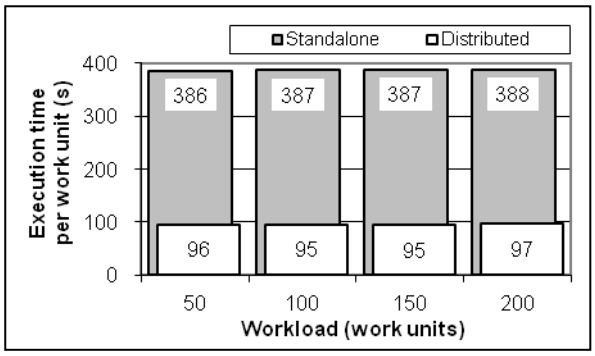

Figure 3: Test results

overnight, when workplace computers are typically idle. For lower priority jobs, daytime background execution may be acceptable.

5.1. Hoarding

Pre-fetching of new work units by the BOINC client has both a positive and negative impact on the operation of the system. By setting the work unit request interval sufficiently short, the client ensures that it has work units in hand before the work unit currently being executed has completed. However, when client computers of differing performance specifications were used on the same application, a phenomenon was observed which we have termed 'hoarding'.

Essentially hoarding occurs because the BOINC system currently provides no fine control over how many work units are pre-fetched by each client, and thus 'fast' clients and 'slow' clients both pre-fetch multiple work units. If the work units are relatively large-grained, the fast clients may 'run dry' before the slow clients have finished processing the first of their work units.

In our experiments, the faster laptops completed around 95% of the total workload and became idle before the slower desktop computers had completed even their first assigned work units. At this point the desktop machines were each hoarding further work units which the laptops could not access, and our initial results showed a total distributed execution time far in excess of the standalone case. The hoarding effect is clearly problematic in this scenario, and we are investigating various methods of enhancing BOINC to compensate for its default behaviour. For these experiments, however, we decided to take measurements from only the four laptops.

Although we experimented with altering the interval between client work unit requests, which reduced the number of work units hoarded by slower machines, we believe that a more sophisticated approach is called for. When a slow client is processing a work unit and a faster client has run dry, the server should be able to send the faster client a duplicate work unit. Although BOINC is able to replicate work units amongst clients to ensure correct results in a PRC context, it does not offer the individual control over work units, or the ability to learn the performance characteristics of each client during operation, that we believe is necessary to fully address the hoarding problem.

6. Conclusions

The work presented here is one of the first attempts to use BOINC in an enterprise desktop grid environment, in the support of the Windows applications commonly found in the workplace.

We have demonstrated that BOINC can speed up the execution of enterprise applications by utilising commodity hardware. In the case of processor-bound applications and dedicated machines, that speedup can approach linear. With different types of application, and where only idle CPU cycles are used, we believe this approach may still offer value, and we intend to pursue experiments along these lines.

We have also demonstrated that Excel may be automated for use in this type of system. Many enterprises have considerable development time and experience with this type of (Windows-based) application, and the ability to distribute workload with only minor modifications is of considerable benefit. Similarly, the utilisation of existing client computers and the minimal investment required on the server side potentially offers the enterprise a strong return on investment in adopting this approach.

However, our experience suggests that there are currently several drawbacks to the use of BOINC in the enterprise. BOINC requires a Linux-based server installation, which may not fit with an enterprise's existing infrastructure or expertise. Creation and management of projects on the BOINC server and operation of the client currently require a degree of intervention from the user, which runs counter to the principle of transparent job processing which we believe desktop grids should aspire to. However, we expect that this burden may be lessened considerably though scripting and automation.

Further, the hoarding issue described in the previous section is a serious impediment for the type of application we expect to encounter in the

enterprise setting. Addressing this may require modification of BOINC itself.

Despite these caveats, and accepting its strong PRC origins, we nonetheless believe that BOINC merits further study as a platform for the development of enterprise desktop grids.

7. Acknowledgements

We acknowledge the assistance of Rahul Talwalkar, Director, Counterparty Risk Management, ING Wholesale Banking; David Cole, Systems Programmer, Brunel University; and the BOINC developers' email list (BOINC_DEV) in the development of this work.

8. References

[1] D. P. Anderson. 'BOINC: A System for Public-Resource Computing and Storage', in *Proceedings of the Fifth International Workshop on Grid Computing (GRID '04)*, Pittsburgh, USA, 2004, pp.4-10.

[2] D. P. Anderson. 'Public Computing: Reconnecting People to Science', in *Proceedings of the Conference on Shared Knowledge and the Web*, Madrid, Spain, 2003.

[3] D. P. Anderson, C. Christensen, and B. Allen. 'Designing a Runtime System for Volunteer Computing', in *Proceedings of the International Conference on High Performance Computing, Networking, Storage, and Analysis (SC2006)*, Tampa, USA, 2006.

[4] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. 'SETI@home: An Experiment in Public-Resource Computing', *Communications of the ACM* 45(11), 2002, pp.56-61.

[5] D. P. Anderson and G. Fedak. 'The Computational and Storage Potential of Volunteer Computing', in *Proceedings of the Sixth International Symposium on Cluster Computing and the Grid (CCGRID 2006)*, Singapore, 2006, pp.73-80.

[6] D. P. Anderson, E. Korpela, and R. Walton. 'High-performance task distribution for volunteer computing', in *Proceedings of the First International Conference on e-Science and Grid Computing*, Melbourne, Australia, 2005, pp.196-203.

[7] A. A. Chien, B. Calder, S. Elbert, and K. Bhatia. 'Entropia: Architecture and performance of an enterprise desktop grid system', *Journal of Parallel and Distributed Computing* 63(5), 2003, pp.597-610.

[8] C. Christensen, T. Aina, and D. Stainforth. 'The challenge of volunteer computing with lengthy climate model simulations', in *Proceedings of the First International Conference on e-Science and Grid Computing (E-SCIENCE '05),* Melbourne, Australia, 2005, pp.8-15.

[9] I. Foster and C. Kesselman. *'The Grid: Blueprint for a New Computing Infrastructure'*, Morgan Kaufmann, United States, ISBN 1558604758, 1998.

[10] I. Foster and A. Iamnitchi. 'On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing', in *Proceedings of the Second International Workshop on Peer-to-Peer Systems (IPTPS 2003)*, Berkley, USA, 2003, pp.118-128.

[11] LHC@home, online at http://lhcathome.cern.ch/ [2007-01-31].

[12] A. Luther, R. Buyya, R. Ranjan, and S. Venugopal. 'Alchemi: A .NET-Based Enterprise Grid Computing System', in *Proceedings of the 6th International Conference on Internet Computing (ICOMP'05)*, Las Vegas, USA, 2005, pp.27-30.

[13] N. Mustafee, A. Alstad, B. Larsen, S. J. E. Taylor, and J. Ladbrook. 'Grid-enabling FIRST: Speeding Up Simulation Applications Using WinGrid', in *Proceedings of the Tenth International Symposium on Distributed Simulation and Real Time Applications (DSRT 2006)*, Malaga, Spain, 2006, pp.157-164.

[14] N. Mustafee and S. J. E. Taylor. 'Using a Desktop Grid to Support Simulation Modelling', in *Proceedings of the 28th Information Technology Interfaces Conference (ITI 2006)*, Dubrovnik, Croatia, 2006, pp.557-562.

[15] D. Stainforth, J. Kettleborough, M. Allen, M. Collins, A. Heaps, and J. Murphy. 'Distributed Computing for Public Interest Climate Modelling Research', *Computing in Science and Engineering* 4(3), 2002, pp.82-89.

[16] M. Taufer, C. An, A. Kerstens, and C. L. Brooks III. 'Predictor@home: A Protein Structure Prediction Supercomputer Based on Global Computing', *IEEE Transactions on Parallel and Distributed Systems* 17(8), 2006, pp.786-796.